MULTI-PROCESSOR SYSTEM VERIFICATION CIRCUITRY

CROSS-REFERENCE TO RELATED APPLICATIONS

Not Applicable

STATEMENT OF FEDERALLY SPONSORED RESEARCH OR DEVELOPMENT

Not Applicable

BACKGROUND OF THE INVENTION

1. TECHNICAL FIELD

This invention relates in general to integrated circuits and, more particularly, to a multiprocessor system verification circuit.

2. DESCRIPTION OF THE RELATED ART

Verifying the operation of an integrated circuit design is generally an extremely complicated procedure. This is especially true in the field of multi-processor designs, where a microprocessor unit is combined with other processor units, such as DSPs (digital signal processors), coprocessors or other microprocessor units.

Verifying the operation of an integrated circuit may take several forms. One type of verification is generally referred to as "design debugging." Debugging techniques are used to resolve weaknesses in the design of an

integrated circuit. Another type of verification is generally referred to as "production testing." The objective of production testing is to identify product which does not meet performance requirements (or to sort product into one of several categories having different production requirements). There are two types of production testing. Functional testing looks for functional differences due to design defects. This type of testing is typically performed using ATPG (automatic test pattern generation) patterns or custom functional test patterns. Performance testing identifies parts that work at a specific speed rating.

Matters are further complicated when design of one of the processor units is from a different source from other processor units on the integrated circuit. This can occur, for example, when a company wishes to combine its microprocessor design with DSPs or coprocessors from another company to provide a specialized integrated circuit. Generally speaking, neither company will want to provide detailed specifications to their designs. Accordingly, verifying the operation of each processor can become problematic.

Therefore, a need has arisen for a method and apparatus for verifying multiprocessor systems.

BRIEF SUMMARY OF THE INVENTION

The present invention provides a processing device comprising a master processor, a system memory and a slave processor subsystem. The slave processor system includes a slave processor, a shared memory accessible by said master processor and said slave processor, and an external memory interface allowing said slave processor to access said system memory. A verification interface passes system memory accesses to the system memory in a normal mode and passes system memory accesses to the shared memory in a verification mode.

The present invention provides significant advantages over the prior art. First, debugging the slave processor subsystem may be performed without understanding the implementation of the master processor subsystem in which the slave processor subsystem is embedded. Second, extraneous interactions are isolated from the slave processor system during verification procedures. Third, the external memory interface can be production tested at operating speed in the same way as an application is actually executed in the field, thereby increasing the fault coverage and capability for performance testing of the slave subsystem.

BRIEF DESCRIPTION OF THE SEVERAL VIEWS OF THE DRAWINGS

For a more complete understanding of the present invention, and the advantages thereof, reference is now made to the following descriptions taken in conjunction with the accompanying drawings, in which:

Figure 1 illustrates a block diagram of a prior art multiprocessor system;

Figure 2 illustrates a block diagram of a multiprocessor system with a verification interface;

Figure 3 illustrates a block diagram of a the verification interface; and

Figure 4 illustrates a block diagram of a multiprocessor system using a master MPU, multiple slave DSP/Coprocessors and verification interface.

DETAILED DESCRIPTION OF THE INVENTION

The present invention is best understood in relation to Figures 1 - 4 of the drawings, like numerals being used for like elements of the various drawings.

Figure 1 illustrates a basic diagram of a multiprocessor system 10 including an MPU subsystem 12 and a DSP/Coprocessor subsystem 14. For purposes of illustration, it will be assumed that the MPU subsystem 12 and the DSP/Coprocessor subsystem 14 are proprietary designs by different companies, although this is not necessary for use of the present invention. MPU subsystem 12 includes a master MPU 16, slave processor boot logic 18 and system memory 20. DSP/Coprocessor subsystem 14 includes an MPU interface 22 for interfacing with the MPU subsystem 12, a shared memory 24 coupled to the MPU interface 22, a slave DSP/Coprocessor 26 coupled to the shared memory 24, a cache memory 28, and an external memory interface 30 coupled to the system memory 20 and cache 28. An optional ROM 32 may be used to store programs or data on the DSP/Coprocessor subsystem 14 for testing purposes.

The block diagram shown in Figure 1 illustrates a general purpose multiprocessor system 10, which could be used for a variety of applications, such as cellular phones, smart phones, personal digital assistants (PDAs), portable computers, and so on. Typically, the DSP/Coprocessor subsystem 14 is used by the multiprocessor system 10 for performing certain tasks, such as voice recognition, handwriting recognition, text-to-speech conversion, to name a few. The DSP/Coprocessor subsystem 14 is designed to execute certain tasks much more efficiently than a general-purpose processor.

Commonly, the designer of a master MPU 16 may wish to combine the MPU subsystem 12 with a DSP/Coprocessor subsystem 14 in order to take advantage of faster execution of certain tasks. During both the design and implementation stages, however, it may be desirable to verify the operation of

the operation of the multiprocessor system 10, including verifying operations of the DSP/Coprocessor subsystem 14.

To debug the DSP/Coprocessor subsystem 14, several steps are necessary. First, programs for execution by the master MPU 16 are loaded into system memory 20. These programs generally comprise code to allow the slave DSP/Coprocessor 26 to function. Second, code for the slave DSP/Coprocessor 26 is loaded into the system memory 20. The master MPU 16 executes the code, setting up the system memory for access by the external memory interface 30 and programming the slave processor boot logic 18 to control the slave processor (s). The slave DSP/Coprocessor 26 programs the cache memory 28 and executes the code from the system memory. The results can be analyzed using traditional debugging techniques, such as setting breakpoints and observing memory locations.

The DSP/Coprocessor subsystem 14 can be difficult to debug within the multiprocessor system 10, since the operation of the MPU subsystem 12 is generally unknown to the designers of the DSP/Coprocessor subsystem 14. Further, the DSP/Coprocessor subsystem 14 is not isolated from extraneous system interactions, such as multiple buses and interfaces on the MPU subsystem 12.

For production performance testing, speed paths in the DSP/Coprocessor subsystem 14 are identified and code is written to activate the speed paths in the DSP/Coprocessor subsystem 14. Code for the master MPU 16 and test patterns and code for the slave DSP/Coprocessor 26 are stored in the system memory 20. The test patterns and slave DSP/Coprocessor code are transferred from the system memory 20 to the shared memory 24. The code is then executed by the slave DSP/Coprocessor 26 to test the external memory interface 30 and the cache 28.

Once again, production testing of parts based on the speed paths in the DSP/Coprocessor subsystem 14 requires knowledge of the detailed operation of the MPU subsystem 12.

In an alternative embodiment, the test pattern and code may be stored in a micro-code ROM 32. In this case, test execution may be initiated by booting to the first address in the micro-code ROM 32.

This embodiment requires a high area overhead for the ROM 32 and has fixed fault coverage.

Figure 2 illustrates a block diagram of a multiprocessor system 40 using a verification interface 42 to aid in debugging and testing. Once again, the multiprocessor system 10 includes an MPU subsystem 12 and a DSP/Coprocessor subsystem 14. MPU subsystem 12 includes a master MPU 16, slave processor boot logic 18 and system memory 20. DSP/Coprocessor subsystem 14 includes an MPU interface 22 for interfacing with the MPU subsystem 12 via the verification interface 42, a shared memory 24 coupled to the MPU interface 22, a slave DSP/Coprocessor 26 coupled to the shared memory 24, a cache memory 28, and an external memory interface 30 coupled to verification interface 42 and cache 28. Verification interface 42 is also coupled to system memory 20. The entire multiprocessor system 40 may be fabricated on a single integrated circuit.

In normal operation of the multiprocessor system 40, the verification interface 42 is disabled. In this state, control and data signals pass between the system memory 20 and external memory interface 30 and between the master MPU 16 and the MPU Interface 22 as shown in Figure 1; i.e., under normal operations, the verification interface 42 is transparent. However, when the verification interface 42 is enabled for verification purposes, requests from the external memory interface 30 to access system memory 20 are translated by the

verification interface 42 such that the shared memory 24 is accessed instead. Hence, the MPU subsystem 12 can be completely isolated from the DSP/Coprocessor subsystem 14 during verification procedures.

The verification interface 42 may be implemented in a independent module of the DSP/Coprocessor subsystem 14 or, alternatively, the verification interface 42 may be implemented as part of the external memory interface 30.

Figure 3 illustrates a block diagram of the verification interface 42. Control signals from the external memory interface 30 to access the system memory 20 are received by demultiplexer 44. When verification mode is disabled, the signals are passed to the system memory 20. When verification mode is enabled, the signals are passed to protocol translator 46. Request multiplexer 46 translates the memory requests to a form acceptable by the MPU interface 22. The output of protocol translator 46 is received by multiplexer 48, which also receives control signals from master MPU 16. When verification mode is disabled, the signals from the master MPU 16 are passed by multiplexer 48 to the MPU interface 22. When verification mode is enabled, multiplexer 48 passes the output of protocol translator 46 to the MPU interface 22. Similarly, multiplexer 50 receives the data from shared memory 24 (via the MPU interface 22) and from system memory 20. When verification mode is disabled, data from the system memory is passed through multiplexer 50 to the external memory interface 30. When verification mode is enabled, data from the shared memory 24 is passed through multiplexer 50 to the external memory interface 30.

During normal operations (i.e., verification mode is disabled), signals pass between the master MPU 16 and the MPU interface 22 and between system memory 20 and the external memory interface 30, as shown in Figure 1. In verification mode, however, the MPU subsystem 12 is isolated from the DSP/Coprocessor subsystem 14. Requests from the external memory interface

30 are translated to a form that is used by the MPU interface 22 to access shared memory 24. Data from the shared memory 24 pursuant to a system memory request is passed to the external memory interface 30.

The protocol translator can translate between different protocol types used by the MPU interface 22 and the external memory interface 30. For example, the external memory interface 30 generally uses a request-based protocol whereas the MPU interface may use a strobe-based protocol. Thus, the protocol translator may translate a request signal to a strobe signal for accessing the shared memory 24 through the MPU interface 22.

Accordingly, referring to Figures 2 and 3, the verification interface 42 can be used to debug the DSP/Coprocessor subsystem 14 without knowledge of the MPU subsystem 12 and to isolate the DSP/Coprocessor subsystem 14 from extraneous system interaction with the MPU subsystem 12. To debug the DSP/Coprocessor subsystem 14, the debug interface programs the external memory interface 30 and loads debug code and data into the shared memory. The verification interface 42 is then enabled in verification mode and the DSP/Coprocessor subsystem 14 is reset. The slave DSP/Coprocessor 26 programs the cache and executes the debug code from shared memory 24. Memory access signals from the external memory interface 30 to system memory are translated by the protocol translator 46, such that the request is fulfilled by shared memory 24. Traditional debugging techniques can then be used to analyze the operation of the DSP/Coprocessor subsystem 14.

For production testing using a critical path test, patterns for activating and testing critical speed paths in the DSP/Coprocessor subsystem 14 can be generated and stored in the shared memory 24. After loading the code in the shared memory 24, the verification interface 42 is enabled and the external memory interface 30 and cache can be tested at speed to determine whether any

of the paths fail. Once again, system memory accesses by the external memory interface 30 are translated by the verification interface 42 and directed to the shared memory 24 via the MPU interface 22. This eliminates the need for a ROM for storing test patterns, saving chip area. Further, the test pattern set can be modified at any time during design or silicon debug, as opposed to a fixed test pattern set encoded in ROM. Since the test pattern set can be changed even after silicon samples are produced, the initial test pattern generation can be easily modified to accommodate late changes in the chip design.

Figure 4 illustrates an embodiment of the invention wherein multiple DSPs and/or coprocessors are implemented. In this case, a system memory arbiter 52 is provided to arbitrate memory requests from the external memory interfaces 30 associated with the various cache memories 28 and slave DSP/Coprocessors 26 (individually referenced as external memory interfaces 30_1 through 30_n, cache memories 28_1 through 28_n and slave DSP/Coprocessors 26_1 through 26_n).

In this embodiment, the verification interface 42 is coupled between a system memory interface 56, including the system memory arbiter 52 and the external memory interfaces 30, and the system memory 20 and between the master MPU 16 and the MPU interface 22. Accesses between various external memory interfaces 30 are resolved by the system memory arbiter 52. Accesses to the shared memory 24 from the master MPU 16 and the slave DSP/Coprocessors 26 are resolved by shared memory interface 54.

If the verification interface 42 is disabled, the requests from the system memory arbiter 52 will be passed to the system memory 20. On the other hand, if the verification interface 42 is enabled for testing or debugging, requests from the system memory arbiter 52 will be translated and passed to the MPU interface 22 for accessing the shared memory 24. This architecture will support any

number of slave DSP/Coprocessors 26. Debugging the subsystem and performing critical path testing can be performed as described above in connection with a multiprocessor system 40 using a single slave DSP/Coprocessor 26.

The present invention provides significant advantages over the prior art in both the debugging and testing of a processor device. First, it is not necessary to understand the implementation of the MPU subsystem 12 in which the DSP/Coprocessor subsystem 14 is embedded in order to debug the DSP/Coprocessor subsystem 14. Second, extraneous system interactions are isolated from the DSP/Coprocessor subsystem 14 during testing. Third, the external memory interface 30 and cache 28 can be production tested at operating speed in the same way as the application is actually executed in the field, thereby increasing the test coverage of the DSP/Coprocessor subsystem 14.

Although the Detailed Description of the invention has been directed to certain exemplary embodiments, various modifications of these embodiments, as well as alternative embodiments, will be suggested to those skilled in the art. The invention encompasses any modifications or alternative embodiments that fall within the scope of the Claims.